MỤC LỤC

TÀI LIỆU HƯỚNG DẪN SỬ DỤNG ANSIBLE TRONG TRIỂN KHAI CI/CD

Tài liệu này bao gồm các nội dung sau:

- *Giới thiệu về các kiến thức chung liên quan đến ansible*
- *Hướng dẫn cài đặt và cấu hình ansible trên server phục vụ cho việc triển khai CI/CD.*
- *Bổ sung một số module hay dùng trong quá trình deploy ứng dụng khi sử dụng ansible.*

I. GIỚI THIỆU CHUNG

Ansible là một công cụ tự động hóa hoàn toàn các hoạt động cung cấp tài nguyên điện toán đám mây, quản lý các thao tác cấu hình, triển khai ứng dụng và quản lý nội dịch vụ.

Được thiết kế để hỗ trợ mô hình triển khai đa pha, Ansible mô hình hóa hạ tầng công nghệ thông tin bằng cách mô tả cách thức các hệ thống dưới góc nhìn tổng quan, thay vì xem xét từng hệ thống một cách riêng lẻ.

Ansible là 1 agent-less IT automation tool được phát triển bởi Michael DeHaan năm 2012. Ansible được tạo ra với mục đích là: minimal, consistent, secure, highly reliable and easy to learn.

Ansible chủ yếu chạy trong chế độ push sử dụng SSH, nghĩa là ta sẽ push các configurations từ server tới các agent. Nhưng ta cũng có thể chạy ansible sử dụng ansible-pull, nghĩa là ta có thể cài đặt ansible lên mỗi agent, sau đó download các playbook từ server về và chạy khi có 1 số lượng lớn các máy tính (số lượng lớn này là bao nhiêu thì tùy thuộc, nhưng ở đây là nhiều hơn 500 máy) và các updates cần thực hiện song song.

Ansible có thể dễ dàng được triển khai bằng cách sử dụng ngôn ngữ YAML để mô tả các công việc tự động hóa.

Phần này giới thiệu về những đặc trưng cơ bản nhất của Ansible.

II. CÁC ĐẶC TRƯNG CƠ BẢN CỦA ANSIBLE

2.1. Kiến trúc

Ansible hoạt động bằng cách kết nối các nodes và đẩy vào các chương trình nhỏ có tên "Ansible modules". Các chương trình này là các mẫu tài nguyên để xây dựng trạng thái mong muốn của cả hệ thống. Ansible sau đó thực thi các modules này (thông qua SSH) và gỡ bỏ nó sau khi hoàn thành.

Thư viện của các modules này có thể nằm trên bất kỳ đâu mà không cần đến một server, daemon hay CSDL nào. Thông thường, bạn chỉ cần thao tác trên terminal, trên một chương trình soạn thảo và một hệ thống version control là đủ.

2.2. Ssh keys

SSH keys là phương thức bảo mật được ưa chuộng bởi Ansible. Module "authorized_key" giúp người dùng quản lý được các machine nào được truy cập host nào. Bên cạnh SSH, các phương thức như kerberos và các hệ thống quản lý danh tính khác cũng có thể được sử dụng.

2.3. Quản lý các Inventory trong một file text đơn giản

Theo mặc định, Ansible mô tả những machines mà nó quản lý sử dụng một file *.ini đơn giản gom chúng chung vào một nhóm. Để có thể thêm một machine mới, ta chỉ việc thêm chúng vào file này. Các dữ liệu từ nguồn khác như EC2, RACKspace hay OpenStack cũng có thể được kéo về sử dụng dynamic inventory.

III. HƯỚNG DẪN CÀI ĐẶT VÀ CẤU HÌNH ANSIBLE

3.1. Cài đặt ansible

Trong tài liệu hướng dẫn cài đặt ansible đơn giản trên Centos 7.

- Bước 1: Install sử dụng yum

  [root@nexus ~]# yum install ansible

- Bước 2: Kiểm tra version ansible đã cài đặt thành công

[root@nexus ~]# ansible --version
ansible 2.8.5
config file = /etc/ansible/ansible.cfg
configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
ansible python module location = /usr/lib/python2.7/site-packages/ansible
executable location = /bin/ansible
python version = 2.7.5 (default, Jun 20 2019, 20:27:34) [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)]

Lưu ý: Ansible kết nối tới server phụ thuộc vào hệ điều hành. Đối với server linux, kết nối thông qua SSH còn server windows kết nối thông qua winrm.

- Bước 3: Cài đặt thêm winrm trên server cài ansible

  pip install pywinrm

Ansible hoạt động sử dụng SSH/WINRM, tức là ta có thể cài đặt nó trên một máy và dùng nó điều khiển cả một hệ thống khác từ xa thông qua SSH. Các phiên bản hiện tại của Ansible có thể hoạt động trên bất kỳ máy nào có cài đặt Python (2.6, 2.7 hoặc >= 3.5). Trên nút điều khiển, ta cần có một phương thức để giao tiếp thông qua ssh, theo mặc định nó có thể là sftp hoặc scp.

Chú ý, trình thông dịch mặc định được sử dụng bởi ansible là /usr/bin/python. Nếu một hệ thống chỉ sử dụng python3, ta có thể gặp phải lỗi giống như sau:

"module_stdout": "/bin/sh: /usr/bin/python: No such file or directory\r\n"

⇨ Ta cần sửa thao số ansible_python_interpreter trỏ vào thư mục /usr/bin/python3 để sử dụng trình thông dịch python3.

Đối với những người bắt đầu tìm hiểu và sử dụng ansible có thể tham khảo tại đây.

3.2. Cấu hình ansible

Host inventory file là file chứa thông tin của các remote server, nơi mà các plays (sẽ được giới thiệu trong phần sau) sẽ được thực thi. Các remote servers (hệ điều hành Linux) phải được cài đặt Python và thư viện simplejson trong trường hợp phiên bản python được cài đặt <= 2.5; các remote server window phải có hệ điều hành Window Server 12 R2 trở lên, Powershell version >=4 và enable winrm để có thể kết nối ansible.

File Ansible configutation có thể được customize để phù hợp với các setting trong môi trường của bạn. Một ansible configuration file sử dụng định .INI để lưu các config. Khi ta chạy 1 ansible command, ansible command đó sẽ tìm file config để thực hiện, tìm theo thứ tự sau:

1. **ANSIBLE_CONFIG**: đầu tiên, Ansible command sẽ kiểm tra các biến môi trường xem có biến môi trường nào trỏ tới file configuration không.

2. **./ansible.cfg**: sau đó, ansible command sẽ tìm xem có file ansible.cfg trong thư mục hiện tại hay không.

3. **~/ansible.cfg**: thứ 3, ansible command sẽ tìm file configuration trong thư mục home của người dùng.

4. **/etc/ansible/ansible.cfg**: cuối cùng, ansible command sẽ kiểm tra file config mặc định đã có khi ta cài đặt Ansible thông qua package manager như pip hay apt/yum.

Trong trường hợp bạn cài ansible thông qua repo trên github thì file **ansible.cfg** sẽ nằm trong thư mục maf bạn đã clone Ansible repository về.

3.3. Cấu hình thông tin của các remote server

Trong một vài trường hợp hiếm gặp khi thiết bị từ xa không hỗ trợ SFTP, ta có thể cấu hình Ansible sử dụng SCP mode thay thế.

Khi giao tiếp với các máy từ xa, Ansible mặc định sử dụng SSH keys. Nếu muốn sử dụng mật khẩu, ta cần thêm các option như --ask-pass hay --ask-become-pass để lấy mật khẩu và mật khẩu root truy cập remote machine.

Để có thể kết nối đến server remote cần đảm bảo thông kết nối từ server cài ansible và server remote.

3.3.1. Làm việc với Inventory file

Ansible có thể làm việc cùng lúc với nhiều hệ thống từ xa với điều kiện chúng được liệt kê trong danh sách Ansible inventory tại vị trí /etc/ansible/hosts.

Không những thế, Ansible còn có thể sử dụng nhiều inventoy files cùng lúc hoặc thậm chí là pull về các dynamic inventory từ nhiều nguồn khác nhau với các định dạng khác nhau (YAML, ini, etc).

- Group là tập hợp của nhiều host trong cùng một hệ thống, ở đây khai báo group dưới dạng một section theo cú pháp:

  # [GROUP_NAME]
  [webserver]
  …..

- Dưới mỗi groupsection là danh sách các hosts, mỗi host nằm trên một dòng, cú pháp khai báo như sau:
  # HOST_NAME[:ALTERNATIVE_SSH_PORT]
  10.60.156.26
- Tạo host alias:
  app1 ansible_ssh_host=10.60.157.89
- Chỉ định người dùng cụ thể để SSH đến:
  HOSTNAME ansible_user=USER_NAME
  10.60.156.26 ansible_user=nexus
- Sử dụng kết nối local thay vì SSH (chạy ansible trên máy trạm):
  localhost ansible_connection=localhost

3.3.2. Cấu hình kết nối ansible với server Linux

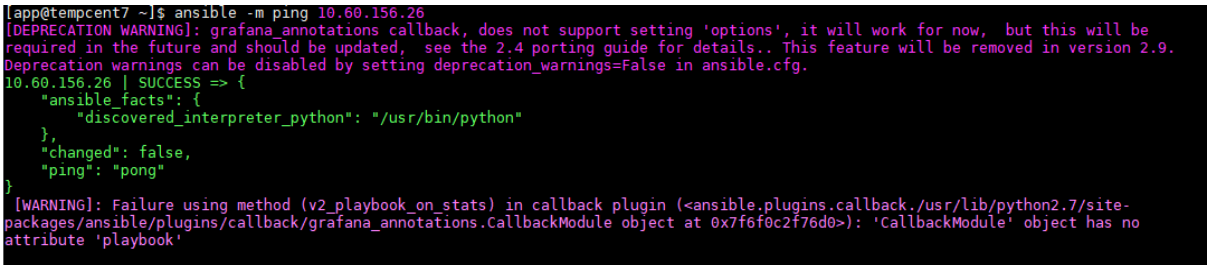
- **Bước 1**: Cấu hình host trong file /etc/ansible/host trên server cài ansible
  Ví dụ:
  [thang10]
  10.60.158.39 ansible_user=smart
  10.60.108.48
  10.60.156.141
- **Bước 2**: Copy key ssh public tới server test
  cat ~/.ssh/id_rsa.pub | ssh smart@**10.60.158.39** 'cat >> ~/.ssh/authorized_keys'

Lưu ý: Trong trường hợp chưa tạo ssh key cần thực hiện các lệnh sau:
ssh-keygen ➔ Tạo public key trên server ansible
Ngoài ra, trong quá trình cài đặt có thể gây ra lỗi **~/.ssh/authorized_keys** không tồn tại, khi đó thực hiện các lệnh sau:

ssh smart@10.60.156.39
mkdir .ssh
cd ~/.ssh
touch authorized_keys

Sau khi chạy xong, trên ansible server thực hiện lại lệnh sau:
cat ~/.ssh/id_rsa.pub | ssh smart@**10.60.158.39** 'cat >> ~/.ssh/authorized_keys'

- Bước 3: Kiểm tra lại kết nối
  Run command trên ansible server:
  ansible -m ping 10.60.156.26



3.3.3. Cấu hình kết nối ansible với server windows

- Bước 1: Config host trong file **/etc/ansible/host** gồm:

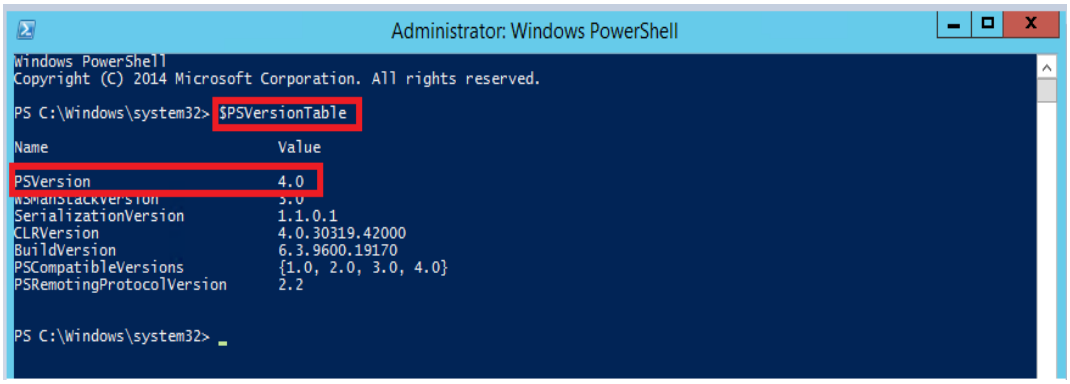
  10.60.156.203 ansible_user=user ansible_password=password ansible_port=10000 ansible_connection=winrm ansible_winrm_server_cert_validation=ignore

  Ngoài ra, có thể tạo file yml trong thư mục /etc/ansible/group_vars/ để cấu hình kết nối:

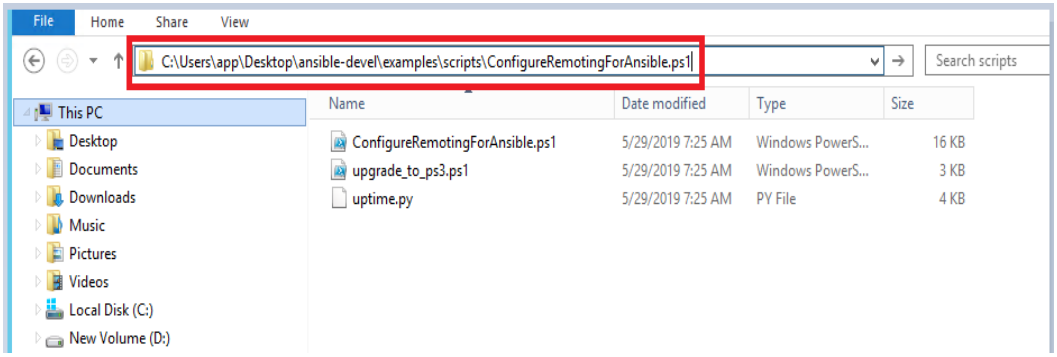
  ansible_user: app_user
  ansible_password: app_password
  ansible_port: 10000
  ansible_connection: winrm
  ansible_winrm_server_cert_validation: ignore
  ansible_winrm_transport: basic
  ansible_winrm_operation_timeout_sec: 60
  ansible_winrm_read_timeout_sec: 70

- Bước 2: Cấu hình winrm trên windows để connect tới ansible như sau:

  1. Kiểm tra powershell để đảm bảo version PS >= 4.0
     - Run PowerShell với quyền Adminitator
     - Run lệnh: $PSVersionTable



  2. Copy file zip ansible-devel.zip vào desktop trên server và giải nén
     - Kiểm tra đường dẫn giải nén để thực hiện cấu hình bước 3



3. Run 2 lệnh sau trong powershell những lệnh sau:
   $file = "C:\Users\app\Desktop\ansible-devel\examples\scripts\ConfigureRemotingForAnsible.ps1"
   powershell.exe -ExecutionPolicy ByPass -File $file –Verbose
4. Thay đổi port winrm https tới 10000 (do xin connect)
   winrm set winrm/config/Listener?Address=*+Transport=HTTP '@{Port="9999"}'
   winrm set winrm/config/Listener?Address=*+Transport=HTTPS '@{Port="10000"}'
5. Set trustHosst tới server jenkins

winrm set winrm/config/client '@{TrustedHosts="10.60.156.96,10.60.156.43"}'

6. Kiểm tra lại config
   winrm e winrm/config/listener

```
PS C:\Windows\system32> winrm enumerate winrm/config/Listener
Listener
    Address = *
    Transport = HTTP
    Port = 9999
    Hostname
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint
    ListeningOn = 10.60.156.203, 127.0.0.1, ::1, fe80::5efe:10.60.156.203%13, fe80::e9e9:e2f1:44e4:e72c%12

Listener
    Address = *
    Transport = HTTPS
    Port = 10000
    Hostname = WIN-T3GLPIK271H
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint = 007EC44495EE3A5A345385B448D1DBEB1AD204FB
    ListeningOn = 10.60.156.203, 127.0.0.1, ::1, fe80::5efe:10.60.156.203%13, fe80::e9e9:e2f1:44e4:e72c%12
```

- Check trên jenkins server. Ví dụ:

```
[app@tempcent7 ~]$ sudo vi /etc/ansible/hosts
[app@tempcent7 ~]$ sudo ansible 10.60.156.203 -m win_ping
10.60.156.203 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[app@tempcent7 ~]$
```

3.4. Ansible playbook

3.4.1. Giới thiệu

Playbooks là ngôn ngữ cấu hình, triển khai và điều phối hệ thống từ xa của Ansible. Nó có thể được dùng để mô tả các chính sách yêu cầu đối với hệ thống từ xa, các bước trong một quy trình IT nào đó hay các bước để deploy một ứng dụng phần mềm.

Trong quy trình triển khai CI/CD của một dự án, ansible được sử dụng để deploy ứng dụng lên nhiều node khác nhau, điều này hỗ trợ nhà phát triển định nghĩa các thao tác cần thực hiện, tránh sai sót trong quá trình triển khai ứng dụng do người dùng thao tác gây ra.

Playbooks được thể hiện bằng cú pháp YAML.

Mỗi playbook chứa một danh sách gồm nhiều 'plays'. Mỗi play có chức năng kết nối một tập các hosts với các vai trò định trước, thao tác này được thể hiện trong các "**tasks**". Mỗi task là một lời gọi đến một ansible module.

Dưới đây là ví dụ về một playbook chỉ chứa một **play**:

```
- hosts: 10.60.156.26
  remote_user: app
  tasks:
  - name: use find to get the files list which you want to copy/fetch
    copy:
      src: "{{ item.src }}"
      dest: "{{ item.dest }}"
    with_items:
      - { src: '/home/app/deploy/a.yml', dest: '/home/app' }
      - { src: '/home/app/deploy/build-impl.xml', dest: '/home/app' }
```

Giải thích: Play này thực hiện trên host có địa chỉ IP là 10.60.156.26 dưới quyền người dùng app, bao gồm các 1 tasks:

Copy nhiều file từ ansible server tới remote server.

Playbook cũng có thể có nhiều **play**:

```
- hosts: 10.60.156.53
  remote_user: app
  tasks:
    - name: Copy file webapp vÃ o thÆ° má»¥c
      copy:
        src: 'build/admin-iist.tar.gz'
        dest: '/home/app/webapp/admin-iist.tar.gz'
    - name: Remove thu muc cu
      shell: rm -rf /home/app/webapp/admin-iist
    - name: Unrar file
      shell: chdir=/home/app/webapp tar -zxf admin-iist.tar.gz
    - name: Rename folder
      command: mv /home/app/webapp/static /home/app/webapp/admin-iist

- hosts: 10.240.192.25
  remote_user: monitor
  tasks:
    - name: Copy file webapp vÃ o thÆ° má»¥c
      copy:
```

```
    src: 'build/admin-iist-public.tar.gz'
    dest: '/home/monitor/webapp/admin-iist.tar.gz'
  - name: Remove thu muc cu
    shell: rm -rf /home/monitor/webapp/admin-iist
  - name: Unrar file
    shell: chdir=/home/monitor/webapp tar -zxf admin-iist.tar.gz
  - name: Rename folder
    command: mv /home/monitor/webapp /home/monitor/webapp/admin-iist
```

3.4.2. Các thành phần trong một playbook

a. *Hosts and Users*

- Chọn các máy trạm và người dùng để chạy một play:

```
- hosts: 10.60.156.53
  remote_user: app
```

- Ta cũng có thể xác định user trong tasks, hoặc chạy play dưới quyền người dùng khác (với các chỉ thị become, become_user, become_method, ...)
- Xác định thứ tự chạy play giữa các hosts:

```
- hosts: all
  order: sorted
  gather_facts: False
  tasks:
    - debug:
        var: inventory_hostname
```

order = {sorted, reverse_sorted (theo tên), inventory, reverse_inventory (theo thứ tự định nghĩa trong file inventory), shuffle (ngẫu nhiên)}

b. Tasks List

Mỗi play gồm một danh sách các tasks. Các tasks được thực thi lần lượt theo thứ tự trên tất cả các máy trong danh sách hosts. Khi thực thi playbook, nếu như một máy nào đó failed với một task nào đó, nó sẽ bị loại ra khỏi vòng lặp thao tác.

Mỗi task có mục đích là thực thi một module nào đó với các tham số cụ thể. Các tham số này có thể được truyền trực tiếp hoặc truyền từ danh sách biến định nghĩa trước.

Các module là **idempotent**, nếu ta đã chạy một module nào đó và đạt được trạng thái mong muốn rồi thì yêu cầu chạy lại nó với danh sách tham số cũ sẽ không được thực hiện.

Mỗi task nên có một tên cụ thể. Tên này được hiển thị ra trong output của playbook. Bởi thế, nên đặt tên dễ hiểu và mô tả chính xác công việc mà task thực hiện.

Mỗi task có một module, mỗi module chứa các tham số dưới dạng dict, duy chỉ có **shell** và **command** là có thể có tham số chứa trong một list.

Sử dụng biến đã định nghĩa trong mục vars:

```
tasks:
  - name: create a virtual host file for {{ vhost }}
    template:
      src: somefile.j2
      dest: /etc/httpd/conf.d/{{ vhost }}
```

c. Thực thi một playbook

- Thực hiện chạy playbook test.yml với command:

ansible-playbook test.yml

- Kết quả chạy playbook:

```
[app@tempcent7 deploy]$ ansible-playbook test.yml

PLAY [10.60.156.26]
*********************************************************************************
*************************************

TASK [Gathering Facts]
*********************************************************************************
*************************************
ok: [10.60.156.26]

TASK [use find to get the files list which you want to copy/fetch]
*********************************************************************
ok: [10.60.156.26] => (item={u'dest': u'/home/app', u'src': u'/home/app/deploy/a.yml'})
ok: [10.60.156.26] => (item={u'dest': u'/home/app', u'src': u'/home/app/deploy/build-
impl.xml'})

PLAY RECAP
*********************************************************************************
*********************************************
10.60.156.26               : ok=2    changed=0    unreachable=0    failed=0    skipped=0   rescued=0
ignored=0
```

IV. MỘT SỐ LỆNH ANSIBLE THƯỜNG DÙNG TRONG QUÁ TRÌNH TRIỂN KHAI CI/CD

4.1. Một số lệnh ansible với linux

a. Copy file or folder from server local to server remote

```
- name: copy file
  copy:
    src: "path_src/file"
    dest: "path_dest/file"
- name: copy folder
  copy:
    src: "path_scr/folder"
    dest:"path_dest/folder"
- name: copy multiple file
  copy:
      src: "{{ item.src }}"
      dest: "{{ item.dest }}"
    with_items:
      - { src: '/home/app/deploy/a.yml', dest: '/home/app' }
      - { src: '/home/app/deploy/build-impl.xml', dest: '/home/app' }
```

b. Copy file from server remote into server local

```
- name: copy file to server remote to workspace jenkins
      fetch:
        src: "path_src/file"
        dest: "path_dest/file"
        flat: yes
```

c. Copy file from server remote to server local use pattern

```
- name: pattern file
      find:
        paths: /home/app/fileresults_HDDT/
        patterns: '*.xls'
      register: files_to_copy
- name: use find to get the files list which you want to copy/fetch
      fetch:
        src: "{{ item.path }}"
        dest: /u02/jenkins/workspace/autoperf_HDDT_test212/resultfiles/
        flat: yes
      with_items: "{{ files_to_copy.files }}"
```

d. Copy 1 file to multi directory in server remote

```
  - name: copy file
        copy:
          src: /home/app/deploy/a.yml
          dest: "{{ item }}"
        with_items:
          - /home/nexus/hddt/aaa.yml
          - /home/nexus/test/bbb.yml
```

e. Run command trong server test để start stop service

```
  - name: Run command
      shell: |
        cd path
        nohup java -jar test.jar &
        ./start.sh
        cd path/tomcat/bin
        nohup ./startup.sh
```

f. Run kill tiến trình

```
  - name: kill process tomcat_autoDeploy
      shell: "ps -ef | grep 'apache-tomcat' | grep -v grep | awk '{print $2}' | xargs -r
  kill -9"
```

g. Xóa file và thư mục trong một thư mục

```
   - name: remove all in folder generated
        file:
          path: /home/app/Voffice/tomcat_autoDeploy/webapps/
          state: "{{ item }}"
        with_items:
          - absent
          - directory
```

h. Check port to start

```
 - name: check port
     shell: |
       cd /home/app/ServerAgent-2.2.1
       flag=`netstat -nap | grep 9000 | grep LISTEN | wc -l`
       if [ $flag -eq 0 ];
       then ./startAgent.sh --udp-port 0 --tcp-port 9000 &
         echo `date`": OK" >> agent_start.log;
       else
         echo `date`": fail, agent is started" >> agent_start.log;
       fi;
```

i. backup file trong linux theo ngày tháng năm

```
- name: backup build artifact module core
    shell: |
      cd /u01/dms_one_domain/ZOTT/builds-7/core-8001
      cp dmscore.core.war dmscore.core.war.bk$(date +%Y%m%d%H%M%S)
```

4.2. Một số lệnh ansible với Window

a. Ansible với windows get artifact into server nexus

```
- name: get file build artifact
    win_get_url:
      url:
http://10.60.156.26:8081/repository/msbuild/YTCS_BUILD_SYS_SERVICE/hison.system/1.{{VERSION}}/hison.system-1.{{VERSION}}.zip
      dest: 'E:\TEST AUTOBUILD\'
```

b. Unzip file windows

```
- name: unzip file build
    win_unzip:
      src: E:\TEST AUTOBUILD\hison.system-1.{{VERSION}}.zip
      dest: E:\TEST AUTOBUILD\SYS\HISONE_SYSTEM\
```

c. Tạo và start website trong IIS

```
- name: deploy app in IIS Server
    win_iis_website:
      name: hison_auto_sys
      state: started
      port: 8188
      ip: 10.60.158.53
      application_pool: HISONE.SYSTEM_DAOTAO
      physical_path: E:\TEST AUTOBUILD\SYS\HISONE_SYSTEM\
```

d. Tao folder backup với ngày tháng năm trong windows

```
- name: Create folder backup
    win_shell: |
      $folderName = (Get-Date).tostring("dd-MM-yyyy-hh-mm-ss")
      New-Item -itemType Directory -Path
C:\S\Apps\4.1\glash\domains\domain1\backup -Name
mtracking_service_api_bk$FolderName
      xcopy
C:\SmartMotor\ps\glfish4.1\glasish\domains\din1\applons\mtrervice_api\*
C:\SmartMotor\Apps\glash4.1\glaish\dains\ain1\bup\mking_service_api_bk$FolderName\ /s /i /Y
```

e. undeploy run command windows

```
- name: undeploy
```

```
      win_command: "{{ item }}"
      with_items:
        - cmd.exe /c C:\aa\aa\aa.1\glassfish\bin\asadmin --port xxxx --host localhost -
u a --passwordfile C:\\Apps\glassfish4.1\glassfish\bin\.txt undeploy
mtracking_service_api
```

f. Xóa sub file và sub folder windows

```
- name: clear cache
      win_file:
        path: C:\Smaror\Apps\gish4.1\sfish\dons\domain1\gated\
        state: "{{ item }}"
      with_items:
        - absent
        - directory
```

g. Copy file windows

```
- name: copy file war to remote server
    win_copy:
     src: target/mtracking_service_api.war
     dest:
C:\SmartMotor\Apps\glassfish4.1\glassfish\domains\domain1\applications\mtracki
ng_service_api.war
  - name: deploy
    win_command: "{{ item }}"
    with_items:
      - cmd.exe /c C:\Smaor\As\glas1\glas\bin\dmin --port xxxx --host localhost -u a
--passwordfile C:\Smaror\Apps\gsh4.1\glish\bin\pas.txt deploy
C:\Smaor\ps\glsh4.1\gish\mns\dain1\aations\mtrvice_api.war
```

h. Backup file windows

```
  - name: backup folder windows
      win_zip:
        src: D:\test
        dest: D:\test_bk_{{ lookup('pipe', 'date +%Y%m%d-%H%M') }}.zip
```